

Cubic Interpolating Splines

CS 257

April 4, 2006

1 Introduction

A function S is a **spline of degree k** if it conforms to the following three conditions.

1. The domain of S is $[a, b]$.
2. S, S', S'', \dots , and $S^{(k-1)}$ are continuous.
3. There are points x_i (the knots of S) such that $a = x_0 < x_1 < \dots < x_n = b$ and such that S is a polynomial of degree k on each subinterval $[x_i, x_{i+1}]$.

Splines have a variety of uses. In CS257, we will deal specifically with the subset of splines that interpolate data points. Assume we have a set of $n + 1$ data points (x_i, y_i) .

x		x_0	...	x_n
y		y_0	...	y_n

An interpolating spline has one additional requirement.

4. $S(x_i) = y_i$ for $i \in [0, n]$

The most common choice of degree is three. The resulting splines are called cubic splines. We use cubic splines for several reasons.

- Because S, S' , and S'' are continuous, these splines appear to be smooth to the eye.
- They are not prohibitively smooth.

- They leave enough “left-over” conditions to be applicable in many cases.
- Odd polynomials have better properties.
- Cubic Splines are best available.

Hence forth, if unqualified the word spline refers to an interpolating cubic spline.

2 Big Picture

We will create a spline $S(x)$ by creating a cubic polynomial $S_i(x)$ for each $[x_i, x_{i+1}]$. We will then piece these n polynomials together in such a way as to conform to the three spline conditions and interpolate the $n + 1$ data points.

$$S(x) = \begin{cases} S_0(x) & x_0 \leq x \leq x_1 \\ S_1(x) & x_1 \leq x \leq x_2 \\ \vdots & \vdots \\ S_{n-1} & x_{n-1} \leq x \leq x_n \end{cases}$$

There are n cubic polynomials. Each polynomial has four coefficients. This results in $4n$ parameters. We use $4n - 2$ of these parameters to force the cubic spline to interpolate the data points and conform to the spline requirements of continuity in the function and its first two derivatives. These conditions break down as follows. Note that the left column sums to $4n - 2$.

Condition	Range	Number of Constraints
$S_i(x_i) = y_i$	$i = 0, \dots, n - 1$	n
$S_i(x_{i+1}) = y_{i+1}$	$i = 0, \dots, n - 1$	n
$S'_i(x_i) = S'_{i+1}(x_i)$	$i = 0, \dots, n - 2$	$n - 1$
$S''_i(x_i) = S''_{i+1}(x_i)$	$i = 0, \dots, n - 2$	$n - 1$

This leaves two free parameters. There are several common choices.

- Fixed slope at the end points

$$\begin{aligned} S'(x_0) &= C_0 \\ S'(x_n) &= C_n \end{aligned}$$

- Natural spline

$$S''(x_0) = S''(x_n) = 0$$

- Not-a-knot condition

S''' is continuous at x_1 and x_{n-1} .

3 Natural Spline Example

Find a natural spline $S(x)$ for the points

$$\begin{array}{c|ccc} x & -1 & 0 & 1 \\ \hline y & 1 & 2 & -1 \end{array}$$

The first step is to write out the spline.

$$S(x) = \begin{cases} S_0(x) = ax^3 + bx^2 + cx + d & x \in [-1, 0] \\ S_1(x) = ex^3 + fx^2 + gx + h & x \in [0, 1] \end{cases}$$

Our next step is to use the spline conditions to find the coefficients a, b, c, d, e, f, g . Requiring S_0 and S_1 to interpolate the data points we find that

$$\begin{aligned} S_0(0) = 2 &\Rightarrow d = 2 \\ S_0(-1) = 1 &\Rightarrow -a + b - c = -1 \\ S_1(0) = 2 &\Rightarrow h = 2 \\ S_1(1) = -1 &\Rightarrow e + f + g = -3 \end{aligned}$$

Requiring continuity of the first derivatives we find

$$S'(x) = \begin{cases} S'_0(x) = 3ax^2 + 2bx + c & x \in [-1, 0] \\ S'_1(x) = 3ex^2 + 2fx + g & x \in [0, 1] \end{cases}$$

$$S'_0(0) = S'_1(0) \Rightarrow c = g$$

Requiring continuity of the second derivatives we find

$$S''(x) = \begin{cases} S''_0(x) = 6ax + 2b & x \in [-1, 0] \\ S''_1(x) = 6ex + 2f & x \in [0, 1] \end{cases}$$

$$S''_0(0) = S''_1(0) \Rightarrow b = f$$

Finally requiring the natural spline conditions to hold we find

$$\begin{aligned} S''_0(-1) = 0 &\Rightarrow 3a = b \\ S''_1(1) = 0 &\Rightarrow 3e = -f \end{aligned}$$

Solving all the linear equations we find that

$$a = -1$$

$$b = -3$$

$$c = -1$$

$$d = 2$$

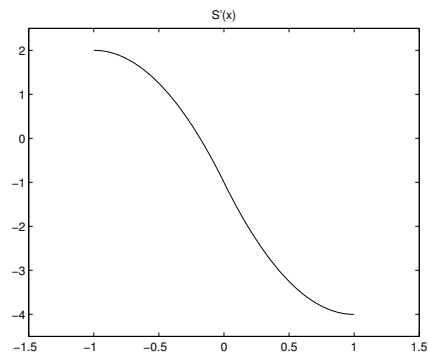
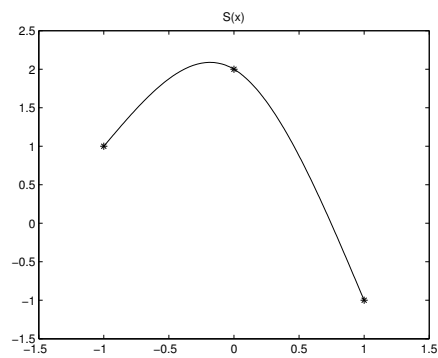
$$e = 1$$

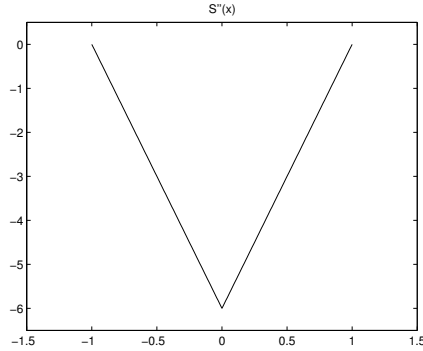
$$f = -3$$

$$g = -1$$

$$h = 2$$

We can plot this and see the results. Note that the first and second derivatives are both continuous and the second derivative is zero at the endpoints.





4 Automation

There are many ways to automate the construction of splines. Here we will diverge from our textbook. Consider again the natural spline

$$S(x) = \begin{cases} S_0(x) & x_0 \leq x \leq x_1 \\ S_1(x) & x_1 \leq x \leq x_2 \\ \vdots & \vdots \\ S_{n-1} & x_{n-1} \leq x \leq x_n \end{cases}$$

$$S(x_i) = y_i$$

$$S'(x) \text{ is continuous}$$

$$S''(x) \text{ is continuous}$$

$$S''(x_0) = S''(x_n) = 0$$

Let $z_i = S''(x_i)$. Because we are using natural splines, this implies $z_0 = z_n = 0$. Suppose now we also know z_i for all $i \in [1, n-1]$. Because S is a set of cubic polynomials with continuous second derivatives, $S''(x)$ is a linear interpolating spline over the points (x_i, z_i) . For example, in the previous section, S'' was a linear spline over the points $(-1, 0)$, $(0, -6)$, $(1, 0)$. We can simply write out one of S'' 's linear functions. Note that S''_i is z_i at the left end point and z_{i+1} at the right.

$$\begin{aligned} S''_i(x) &= \frac{z_{i+1}}{x_{i+1}-x_i}(x-x_i) + \frac{z_i}{x_{i+1}-x_i}(x_{i+1}-x) \\ &= \frac{z_{i+1}}{h_i}(x-x_i) + \frac{z_i}{h_i}(x_{i+1}-x) \end{aligned}$$

Integrating twice we find

$$\begin{aligned} S_i(x) &= \frac{z_{i+1}}{6h_i}(x-x_i)^3 + \frac{z_i}{6h_i}(x_{i+1}-x)^3 + cx + d \\ &= \frac{z_{i+1}}{6h_i}(x-x_i)^3 + \frac{z_i}{6h_i}(x_{i+1}-x)^3 + c_i(x-x_i) + d_i(x_{i+1}-x) \end{aligned}$$

The first and last equation can be eliminated to get a tridiagonal system of order $n - 1$.

$$\begin{bmatrix} u_1 & h_1 & & & & \\ h_1 & u_2 & h_2 & & & \\ & & \ddots & \ddots & & \\ & & & h_{n-3} & u_{n-2} & h_{n-2} \\ & & & & h_{n-2} & u_{n-1} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_{n-1} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-1} \end{bmatrix}$$

One way to solve the system is to use Gaussian Elimination without pivoting.

1. Forward substitution. For $i = 2, 3, \dots, n - 1$

$$\begin{aligned} u_i &\leftarrow u_i - \frac{h_{i-1}^2}{u_{i-1}} \\ v_i &\leftarrow v_i - \frac{h_{i-1}v_{i-1}}{u_{i-1}} \end{aligned}$$

2. Backward substitution. First assign z_{n-1}

$$z_{n-1} \leftarrow \frac{v_{n-1}}{u_{n-1}}$$

Next assign the rest. For $i = n - 2, n - 3, \dots, 1$

$$z_i \leftarrow \frac{v_i - h_i z_{i+1}}{u_i}$$

5 Algorithm

Input: $n + 1$ interpolation points $(x_0, y_0), \dots, (x_n, y_n)$

Output: Cubic interpolating spline $S(x)$

1. For $i = 0, 1, \dots, n - 1$ compute

$$\begin{aligned} h_i &= x_{i+1} - x_i \\ b_i &= \frac{y_{i+1} - y_i}{h_i} \end{aligned}$$

2. Set

$$\begin{aligned} u_1 &= 2(h_0 + h_1) \\ v_1 &= 6(b_1 - b_0) \end{aligned}$$

and then compute for $i = 2, 3, \dots, n - 1$

$$\begin{aligned} u_i &= 2(h_i + h_{i+1}) - \frac{h_{i-1}^2}{u_{i-1}} \\ v_i &= 6(b_i - b_{i-1}) - \frac{h_{i-1}v_{i-1}}{u_{i-1}} \end{aligned}$$

3. Set

$$\begin{aligned} z_0 &= 0 \\ z_n &= 0 \end{aligned}$$

and then compute for $i = n - 1, n - 2, \dots, 1$

$$z_i = \frac{v_i - h_i z_{i+1}}{u_i}$$

4. Substitutes all the coefficients into the S_i 's.

$$\begin{aligned} S_i(x) &= \frac{z_{i+1}}{6h_i}(x - x_i)^3 + \frac{z_i}{6h_i}(x_{i+1} - x)^3 \\ &\quad + \left(\frac{y_{i+1}}{h_i} - \frac{z_{i+1}}{6}h_i \right) (x - x_i) + \left(\frac{y_i}{h_i} - \frac{z_i}{6}h_i \right) (x_{i+1} - x) \end{aligned}$$

It can be shown that for all i , u_i is never zero. The algorithm cannot fail.

6 Nested Evaluation

The above form is hard to work with. What we really want is to write

$$S_i(x) = A_i + B_i(x - x_i) + C_i(x - x_i)^2 + D_i(x - x_i)^3$$

This is the Taylor series of S_i about x_i .

$$A_i = S_i(x_i) \quad B_i = S_i'(x_i) \quad C_i = \frac{1}{2}S_i''(x_i) \quad D_i = \frac{1}{6}S_i'''(x_i)$$

By the definition of S_i ,

$$A_i = y_i$$

Remembering that $z_i = S_i'''(x_i)$,

$$C_i = \frac{z_i}{2}$$

Working through the rest we find,

$$B_i = -\frac{h_i}{6}z_{i+1} - \frac{h_i}{3}z_i + \frac{y_{i+1} - y_i}{h_i}, \quad D_i = \frac{z_{i+1} - z_i}{6h_i}$$

Finally, we put the polynomial into nested form

$$S_i(x) = A_i + (x - x_i)(B_i + (x - x_i)(C_i + (x - x_i)D_i))$$

7 References

For more complete pseudocode of the above algorithm, see Kincaid and Cheney's *Numerical Mathematics and Computing*, fifth edition, pages 404-406.

For more information on the MATLAB built-in spline functionality see Recktenwald, page 584